DRAFT − CUPS Software Programmers Manual

CUPS−SPM−1.1

Easy Software Products

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Preface

This software programmers manual provides software programming information for the Common UNIX Printing System ("CUPS") Version 1.1.

System Overview

The Common UNIX Printing System provides a portable printing layer for UNIX® operating systems. It has been developed by Easy Software Products to promote a standard printing solution for all UNIX vendors and users. CUPS provides the System V and Berkeley command−line interfaces.

CUPS uses the Internet Printing Protocol (IETF−IPP) as the basis for managing print jobs and queues. The Line Printer Daemon (LPD, RFC1179), Server Message Block (SMB), and AppSocket protocols are also supported with reduced functionality.

CUPS adds network printer browsing and PostScript Printer Description ("PPD")−based printing options to support real world applications under UNIX.

CUPS also includes a customized version of GNU GhostScript (currently based off GNU GhostScript 5.50) and an image file RIP that is used to support non−PostScript printers.

Document Overview

This software administrators manual is organized into the following sections:

- 1 − Printing System Overview

- 2 − The CUPS API
- 3 − Writing Filters
- 4 − Writing Printer Drivers
- 5 − Writing Backends
- A − Constants
- B − Structures
- C − Functions

1 − Printing System Overview

This chapter provides an overview of how the Common UNIX Printing System works.

The Printing Problem

For years *the printing problem* has plagued UNIX®. Unlike Microsoft® Windows® or MacOS, UNIX has no standard interface or system in place for supporting printers. Among the solutions previously available, the Berkeley and System V printing systems are the most prevalent.

These printing systems support line printers (text only) or PostScript printers (text and graphics), and with some coaxing they can be made to support a full range of printers and file formats. However, because each varient of the UNIX operating system uses a different printing system than the next, developing printer drivers for a wide range of printers is extremely difficult. That combined with the limited volume of customers for each UNIX varient has forced most printer vendors to give up supporting UNIX entirely.

The Common UNIX Printing System, or CUPS, is designed to eliminate *the printing problem*. One common printing system can be used by all UNIX varients to support the printing needs of users. Printer vendors can use its modular filter interface to develop a single driver program that supports a wide range of file formats with little or no effort. Since CUPS provides both the System V and Berkeley printing commands, users (and applications) can reap the benefits of this new technology with no changes.

The Technology

CUPS is based upon an emerging Internet standard called the Internet Printing Protocol, or IPP. IPP has been embraced by dozens of printer and printer server manufacturers, and will be supported by the next Microsoft Windows operating system.

IPP defines a standard protocol for printing as well as managing print jobs and printer options like media size, resolution, and so forth. Like all IP−based protocols, IPP can be used locally or over the Internet to printers hundreds or thousands of miles away. Unlike other protocols, however, IPP also supports access control, authentication, and encryption, making it a much more secure printing solution than older ones.

IPP is layered on top of the Hyper−Text Transport Protocol, or HTTP, which is the basis of web servers on the Internet. This allows the user to view documentation and status information on a printer or server using their web browser.

CUPS provides a complete IPP/1.0−based printing system that provides Basic authentication and domain or IP−based access control. Digest authentication and TLS encryption will be available in future versions of CUPS.

Jobs

Each file that is submitted for printing is called a *job*. Jobs are identified by a unique number starting at 1 and are assigned to a particular destination (usually a printer). Jobs can also have options associated with them such as media size, number of copies, and priority.

Classes

CUPS supports collections of printers known as *classes*. Jobs sent to a class are forwarded to the first available printer in the class.

Filters

Filters allow a user or application to print many types of files without extra effort. Print jobs sent to a CUPS server are filtered before sending them to a printer. Some filters convert job files to different formats that the printer can understand. Others perform page selection and ordering tasks. *Backend* filters perform the most important task of all − they send the filtered print data to the printer.

CUPS provides filters for printing many types of image files, HP−GL/2 files, PDF files, and text files. CUPS also supplies PostScript and image file Raster Image Processors, or RIPs, that convert PostScript or image files into bitmaps that can be sent to a raster printer.

CUPS provides backends for printing over parallel and serial ports, and over the network via the JetDirect (AppSocket), Server Message Block, and Line Printer Daemon protocols.

Printer Drivers

Printer drivers in CUPS consist of one of more filters specific to a printer. CUPS includes a sample printer driver for Hewlett−Packard LaserJet and DeskJet printers. While this driver does not generate optimal output for different printer models, it does demonstrate how you can write your own printer drivers and incorporate them into CUPS.

Networking

Printers and classes on the local system are automatically shared with other systems on the network. This allows you to setup one system to print to a printer and use this system as a printer server or spool host for all of the others. If there is only one occurrence of a printer on a network, then that printer can be accessed using its name alone. If more than one printer exists with the same name, users must select the printer by specifying which server to use (e.g. "printer@host1" or "printer@host2".)

CUPS also provides *implicit classes*, which are collections of printers and/or classes with the same name. This allows you to setup multiple servers pointing to the same physical network printer, for example, so that you aren't relying on a single system for printing. Because this also works with printer classes, you can setup multiple servers and printers and never worry about a "single point of failure" unless all of the printers and servers goes down!

2 − The CUPS API

This chapter describes the CUPS Application Programmers Interface ("API").

The CUPS Library

Detecting the CUPS Library in Autoconf

Basic Services

Include Files

Getting the Available Printers and Classes

Printing Files

Setting Printer Options

Cancelling Jobs

HTTP Services

Include Files

Connecting to a Server

Setting Request Fields

Issuing a Request

Getting the Request Status

Sending Request Data

Reading Request Data

IPP Services

Include Files

Creating an IPP Request

Adding Attributes

Sending an IPP Request

Reading an IPP Response

Finding Attributes

Looping Through Attributes

IPP Standard Operations

IPP Extension Operations

CUPS Extension Operations

Language Services

Include Files

Getting the Default Language

Getting the Language Encoding

Getting a Language String

PPD Services

Include Files

Loading a PPD File

Options and Groups

Finding an Option

Finding a Page Size

Marking Options

Checking for Conflicts

Sending Options

3 − Writing Filters

This chapter describes how to write a file filter for CUPS.

Overview

Security Considerations

Users and Groups

Temporary Files

Page Accounting

Command−Line Arguments

Copy Generation

Environment Variables

Writing a HTML Filter

4 − Writing Printer Drivers

This chapter discusses how to write a printer driver, which is a special filter program that converts CUPS raster data into the appropriate commands and data required for a printer.

Overview

Page Accounting

Color Management

Raster Functions

cupsRasterOpen()

cupsRasterReadHeader()

cupsRasterReadPixels()

cupsRasterClose()

Writing a HP−PCL Driver

5 – Writing Backends

This chapter describes how to write a backend for CUPS. Backends communicate directly with printers and allow printer drivers and filters to send data using any type of connection transparently.

Overview

Security Considerations

Users and Groups

Temporary Files

Page Accounting

Retries

Command−Line Arguments

Copy Generation

Environment Variables

Writing a Serial Port Backend

A − Constants

This appendix lists all of the constants that are defined by the CUPS API.

CUPS Constants

HTTP Constants

IPP Constants

Language Constants

PPD Constants

Raster Constants

B − Structures

This appendix describes all of the structures that are defined by the CUPS API.

C − Functions

This appendix provides a reference for all of the CUPS API functions.

cupsAddOption()

Usage

```
int
cupsAddOption(const char *name,
              const char *value,
              int num_options,
              cups_option_t **options);
```

Arguments

Argument	Description
name	The name of the option.
value	The value of the option.
num_options	Number of options currently in the array.
options	Pointer to the options array.

Returns

The new number of options.

Description

`cupsAddOption()` adds an option to the specified array.

Example

```
#include <cups.h>

...

/* Declare the options array */
int           num_options;
cups_option_t *options;

/* Initialize the options array */
num_options = 0;
options     = (cups_option_t *)0;

/* Add options using cupsAddOption() */
num_options = cupsAddOption("media", "letter", num_options, &options);
num_options = cupsAddOption("resolution", "300dpi", num_options, &options);
```

See Also

`cupsFreeOptions()`, `cupsGetOption()`, `cupsParseOptions()`

cupsCancelJob()

Usage

```
int
cupsCancelJob(const char *dest,
              int job);
```

Arguments

Argument	Description
dest	Printer or class name
job	Job ID

Returns

1 on success, 0 on failure. On failure the error can be found by calling `cupsLastError()`.

Description

`cupsCancelJob()` cancels the specifies job.

Example

```
#include <cups.h>

cupsCancelJob("LaserJet", 1);
```

See Also

`cupsLastError()`, `cupsPrintFile()`

cupsDoFileRequest()

Usage

```
ipp_t *
cupsDoFileRequest(http_t *http,
                  ipp_t *request,
                  const char *resource,
                  const char *filename);
```

Arguments

Argument	Description
http	HTTP connection to server.
request	IPP request data.
resource	HTTP resource name for POST.
filename	File to send with POST request (`NULL` pointer if none.)

Returns

IPP response data or `NULL` if the request fails. On failure the error can be found by calling `cupsLastError()`.

Description

`cupsDoFileRequest()` does a HTTP POST request and provides the IPP request and optionally the contents of a file to the IPP server. It also handles resubmitting the request and performing password authentication as needed.

Example

```
#include <cups.h>

http_t       *http;
cups_lang_t *language;
ipp_t        *request;
ipp_t        *response;

...

/* Get the default language */
language = cupsLangDefault();

/* Create a new IPP request */
request  = ippNew();

request->request.op.operation_id = IPP_PRINT_FILE;
```

```
request->request.op.request_id   = 1;

/* Add required attributes */
ippAddString(request, IPP_TAG_OPERATION, IPP_TAG_CHARSET,
             "attributes-charset", NULL, cupsLangEncoding(language));

ippAddString(request, IPP_TAG_OPERATION, IPP_TAG_LANGUAGE,
             "attributes-natural-language", NULL,
             language != NULL ? language->language : "C");

ippAddString(request, IPP_TAG_OPERATION, IPP_TAG_URI, "printer-uri",
             NULL, "ipp://hostname/resource");

ippAddString(request, IPP_TAG_OPERATION, IPP_TAG_NAME, "requesting-user-name",
             NULL, cupsUser());

/* Do the request... */
response = cupsDoFileRequest(http, request, "/resource", "filename.txt");
```

See Also

cupsLangDefault(), cupsLangEncoding(), cupsUser(), httpConnect(), ippAddString(), ippNew()

cupsDoRequest()

Usage

```
ipp_t *
cupsDoRequest(http_t *http,
              ipp_t *request,
              const char *resource);
```

Arguments

Argument	Description
http	HTTP connection to server.
request	IPP request data.
resource	HTTP resource name for POST.

Returns

IPP response data or `NULL` if the request fails. On failure the error can be found by calling `cupsLastError()`.

Description

`cupsDoRequest()` does a HTTP POST request and provides the IPP request to the IPP server. It also handles resubmitting the request and performing password authentication as needed.

Example

```
#include <cups.h>

http_t      *http;
cups_lang_t *language;
ipp_t       *request;
ipp_t       *response;

...

/* Get the default language */
language = cupsLangDefault();

/* Create a new IPP request */
request  = ippNew();

request->request.op.operation_id = IPP_GET_PRINTER_ATTRIBUTES;
request->request.op.request_id   = 1;

/* Add required attributes */
ippAddString(request, IPP_TAG_OPERATION, IPP_TAG_CHARSET,
             "attributes−charset", NULL, cupsLangEncoding(language));

ippAddString(request, IPP_TAG_OPERATION, IPP_TAG_LANGUAGE,
             "attributes−natural−language", NULL,
             language != NULL ? language−>language : "C");

ippAddString(request, IPP_TAG_OPERATION, IPP_TAG_URI, "printer−uri",
             NULL, "ipp://hostname/resource");

/* Do the request... */
response = cupsDoRequest(http, request, "/resource");
```

See Also

cupsLangDefault(), cupsLangEncoding(), cupsUser(), httpConnect(), ippAddString(), ippNew()

cupsFreeOptions()

Usage

```
void
cupsFreeOptions(int num_options,
                cups_option_t *options);
```

Arguments

Argument	Description
num_options	Number of options in array.
options	Pointer to options array.

Description

`cupsFreeOptions()` frees all memory associated with the option array specified.

Example

```
#include <cups/cups.h>

int           num_options;
cups_option_t *options;

...

cupsFreeOptions(num_options, options);
```

See Also

cupsAddOption(), cupsGetOption(), cupsMarkOptions(), cupsParseOptions()

cupsGetClasses()

Usage

```
int
cupsGetClasses(char ***classes);
```

Arguments

Argument	Description
classes	Pointer to character pointer array.

Returns

The number of printer classes available.

Description

`cupsGetClasses()` gets a list of the available printer classes. The returned array should be freed using the `free()` when it is no longer needed.

Example

```
#include <cups/cups.h>

int  i;
int  num_classes;
char **classes;

...

num_classes = cupsGetClasses(;

...

if (num_classes > 0)
{
  for (i = 0; i num_classes; i ++)
    free(classes[i]);

  free(classes);
}
```

See Also

cupsGetDefault(), cupsGetPrinters()

cupsGetDefault()

Usage

```
const char *
cupsGetDefault(void);
```

Returns

A pointer to the default destination.

Description

`cupsGetDefault()` gets the default destination printer or class. The default destination is stored in a static string and will be overwritten (usually with the same value) after each call.

Example

```
#include <cups/cups.h>

printf("The default destination is %s\n", cupsGetDefault());
```

See Also

cupsGetClasses(), cupsGetPrinters()

cupsGetOption()

Usage

```
const char *
cupsGetOption(const char *name,
              int num_options,
              cups_option_t *options);
```

Arguments

Argument	Description
name	The name of the option.
num_options	The number of options in the array.
options	The options array.

Returns

A pointer to the option values or `NULL` if the option is not defined.

Description

`cupsGetOption()` returns the first occurrence of the named option. If the option is not included in the options array then a `NULL` pointer is returned.

```
#include <cups/cups.h>

int           num_options;
cups_option_t *options;
const char    *media;

...

media = cupsGetOption("media", num_options, options);
```

See Also

cupsAddOption(), cupsFreeOptions(), cupsMarkOptions(), cupsParseOptions()

cupsGetPassword()

Usage

```
const char *
cupsGetPassword(const char *prompt);
```

Arguments

Argument	Description
prompt	The prompt to display to the user.

Returns

A pointer to the password that was entered or `NULL` if no password was entered.

Description

`cupsGetPassword()` displays the prompt string and asks the user for a password. The password text is not echoed to the user.

Example

```
#include <cups/cups.h>

char *password;

...

password = cupsGetPassword("Please enter a password:");
```

See Also

cupsServer(), cupsUser()

cupsGetPPD()

Usage

```
const char *
cupsGetPPD(const char *printer);
```

Arguments

Argument	Description
printer	The name of the printer.

Returns

The name of a temporary file containing the PPD file or `NULL` if the printer cannot be located or does not have a PPD file.

Description

`cupsGetPPD()` gets a copy of the PPD file for the named printer. The printer name can be of the form "printer" or "printer@hostname".

You should remove (unlink) the PPD file after you are done using it. The filename is stored in a static buffer and will be overwritten with each call to `cupsGetPPD()`.

Example

```
#include <cups/cups.h>

char *ppd;

...

ppd = cupsGetPPD("printer@hostname");

...

unlink(ppd);
```

cupsGetPrinters()

Usage

```
int
cupsGetPrinters(char ***printers);
```

Arguments

Argument	Description
printers	Pointer to character pointer array.

Returns

The number of printer printers available.

Description

`cupsGetPrinters()` gets a list of the available printers. The returned array should be freed using the `free()` when it is no longer needed.

Example

```
#include <cups/cups.h>

int   i;
int   num_printers;
char **printers;

...

num_printers = cupsGetPrinters(;

...

if (num_printers > 0)
{
  for (i = 0; i num_printers; i ++)
    free(printers[i]);

  free(printers);
}
```

See Also

cupsGetClasses(), cupsGetDefault()

cupsLangDefault()

Usage

```
const char *
cupsLangDefault(void);
```

Returns

A pointer to the default language structure.

Description

`cupsLangDefault()` returns a language structure for the default language. The default language is defined by the `LANG` environment variable. If the specified language cannot be located then the POSIX (English) locale is used.

Call `cupsLangFree()` to free any memory associated with the language structure when you are done.

Example

```
#include <cups/language.h>

cups_lang_t *language;
...

language = cupsLangDefault();

...

cupsLangFree(language);
```

See Also

cupsLangEncoding(), cupsLangFlush(), cupsLangFree(), cupsLangGet(), cupsLangString()

cupsLangEncoding()

Usage

```
char *
cupsLangEncoding(cups_lang_t *language);
```

Arguments

Argument	Description
language	The language structure.

Returns

A pointer to the encoding string.

Description

`cupsLangEncoding()` returns the language encoding used for the specified language, e.g. "iso−8859−1", "utf−8", etc.

Example

```
#include <cups/language.h>

cups_lang_t *language;
char        *encoding;
...

language = cupsLangDefault();
encoding = cupsLangEncoding(language);
...

cupsLangFree(language);
```

See Also

cupsLangDefault(), cupsLangFlush(), cupsLangFree(), cupsLangGet(), cupsLangString()

cupsLangFlush()

Usage

```
void
cupsLangFlush(void);
```

Description

`cupsLangFlush()` frees all language structures that have been allocated.

Example

```
#include <cups/language.h>

...

cupsLangFlush();
```

See Also

cupsLangDefault(), cupsLangEncoding(), cupsLangFree(), cupsLangGet(), cupsLangString()

cupsLangFree()

Usage

```
void
cupsLangFree(cups_lang_t *language);
```

Arguments

Argument	Description
language	The language structure to free.

Description

`cupsLangFree()` frees the specified language structure.

Example

```
#include <cups/language.h>

cups_lang_t *language;
...

cupsLangFree(language);
```

See Also

cupsLangDefault(), cupsLangEncoding(), cupsLangFlush(), cupsLangGet(), cupsLangString()

cupsLangGet()

Usage

```
cups_lang_t *
cupsLangGet(const char *name);
```

Arguments

Argument	Description
name	The name of the locale.

Returns

A pointer to a language structure.

Description

`cupsLangGet()` returns a language structure for the specified locale. If the locale is not defined then the POSIX (English) locale is substituted.

Example

```
#include <cups/language.h>

cups_lang_t *language;

...

language = cupsLangGet("fr");

...

cupsLangFree(language);
```

See Also

cupsLangDefault(), cupsLangEncoding(), cupsLangFlush(), cupsLangFree(), cupsLangString()

cupsLangString()

Usage

```
char *
cupsLangString(cups_lang_t *language,
               int          message);
```

Arguments

Argument	Description
language	The language to query.
message	The message number.

Returns

A pointer to the message string or `NULL` if the message is not defined.

Description

`cupsLangString()` returns a pointer to the specified message string in the specified language.

Example

```
#include <cups/language.h>

cups_lang_t *language;
char        *s;
...

language = cupsLangGet("fr");

s = cupsLangString(language, CUPS_MSG_YES);

...

cupsLangFree(language);
```

See Also

cupsLangDefault(), cupsLangEncoding(), cupsLangFlush(), cupsLangFree(), cupsLangGet()

cupsLastError()

Usage

```
ipp_status_t
cupsLastError(void);
```

Returns

An enumeration containing the last IPP error.

Description

`cupsLastError()` returns the last IPP error that occurred. If no error occurred then it will return `IPP_OK` or `IPP_OK_CONFLICT`.

Example

```
#include <cups/cups.h>

ipp_status_t status;

...

status = cupsLastError();
```

See Also

cupsCancelJob(), cupsPrintFile()

cupsMarkOptions()

Usage

```
int
cupsMarkOptions(ppd_file_t *ppd,
                int num_options,
                cups_option_t *options);
```

Arguments

Argument	Description
ppd	The PPD file to mark.
num_options	The number of options in the options array.
options	A pointer to the options array.

Returns

The number of conflicts found.

Description

`cupsMarkOptions()` marks options in the PPD file. It also handles mapping of IPP option names and values to PPD option names.

Example

```
#include <cups/cups.h>

int           num_options;
cups_option_t *options;
ppd_file_t    *ppd;

...

cupsMarkOptions(ppd, num_options, options);
```

See Also

cupsAddOption(), cupsFreeOptions(), cupsGetOption(), cupsParseOptions()

cupsParseOptions()

Usage

```
int
cupsParseOptions(const char *arg,
                 int num_options,
                 cups_option_t **options);
```

Arguments

Argument	Description
arg	The string containing one or more options.
num_options	The number of options in the options array.
options	A pointer to the options array pointer.

Returns

The new number of options in the array.

Description

`cupsParseOptions()` parses the specifies string for one or more options of the form "name=value", "name", or "noname". It can be called multiple times to combine the options from several strings.

Example

```
#include <cups/cups.h>

int           num_options;
cups_option_t *options;

...

num_options = 0;
options     = (cups_option_t *)0;
num_options = cupsParseOptions(argv[5], num_options, &options);
```

See Also

cupsAddOption(), cupsFreeOptions(), cupsGetOption(), cupsMarkOptions()

cupsPrintFile()

Usage

```
int
cupsPrintFile(const char *printer,
              const char *filename,
              const char *title,
              int num_options,
              cups_option_t *options);
```

Arguments

Argument	Description
printer	The printer or class to print to.
filename	The file to print.
title	The job title.
num_options	The number of options in the options array.
options	A pointer to the options array.

Returns

The new job ID number or 0 on error.

Description

`cupsPrintFile()` sends a file to the specified printer or class for printing. If the job cannot be printed the error code can be found by calling `cupsLastError()`.

Example

```
#include <cups/cups.h>

int           num_options;
cups_option_t *options;

...

cupsPrintFile("printer@hostname", "filename.ps", "Job Title", num_options,
              options);
```

See Also

cupsCancelJob(), cupsLastError()

cupsRasterClose()

Usage

```
void
cupsRasterClose(cups_raster_t *ras);
```

Arguments

Argument	Description
ras	The raster stream to close.

Description

`cupsRasterClose()` closes the specified raster stream.

Example

```
#include <cups/raster.h>

cups_raster_t *ras;

...

cupsRasterClose(ras);
```

See Also

cupsRasterOpen(), cupsRasterReadHeader(), cupsRasterReadPixels(), cupsRasterWriteHeader(), cupsRasterWritePixels()

cupsRasterOpen()

Usage

```
cups_raster_t *
cupsRasterOpen(int fd,
               cups_mode_t mode);
```

Arguments

Argument	Description
fd	The file descriptor to use.
mode	The mode to use; `CUPS_RASTER_READ` or `CUPS_RASTER_WRITE`.

Returns

A pointer to a raster stream or `NULL` if there was an error.

Description

`cupsRasterOpen()` opens a raster stream for reading or writing.

Example

```
#include <cups/raster.h>

cups_raster_t *ras;

...

ras = cupsRasterOpen(0, CUPS_RASTER_READ);
```

See Also

cupsRasterClose(), cupsRasterReadHeader(), cupsRasterReadPixels(), cupsRasterWriteHeader(), cupsRasterWritePixels()

cupsRasterReadHeader()

Usage

```
unsigned
cupsRasterReadHeader(cups_raster_t *ras,
                     cups_page_header_t *header);
```

Arguments

Argument	Description
ras	The raster stream to read from.
header	A pointer to a page header structure to read into.

Returns

1 on success, 0 on EOF or error.

Description

`cupsRasterReadHeader()` reads a page header from the specified raster stream.

Example

```
#include <cups/raster.h>

int                   line;
cups_raster_t        *ras;
cups_raster_header_t header;
unsigned char        pixels[8192];
...

while (cupsRasterReadHeader(ras, &header))
{
  ...

  for (line = 0; line < header.cupsHeight; line ++)
  {
    cupsRasterReadPixels(ras, pixels, header.cupsBytesPerLine);

    ...
  }
}
```

See Also

cupsRasterClose(), cupsRasterOpen(), cupsRasterReadPixels(), cupsRasterWriteHeader(), cupsRasterWritePixels()

cupsRasterReadPixels()

Usage

```
unsigned
cupsRasterReadPixels(cups_raster_t *ras,
                     unsigned char *pixels,
                     unsigned length);
```

Arguments

Argument	Description
ras	The raster stream to read from.
pixels	The pointer to a pixel buffer.
length	The number of bytes of pixel data to read.

Returns

The number of bytes read or 0 on EOF or error.

Description

`cupsRasterReadPixels()` reads pixel data from the specified raster stream.

Example

```
#include <cups/raster.h>

int                   line;
cups_raster_t         *ras;
cups_raster_header_t  header;
unsigned char         pixels[8192];
...

while (cupsRasterReadHeader(ras, &header))
{
  ...

  for (line = 0; line < header.cupsHeight; line ++)
  {
    cupsRasterReadPixels(ras, pixels, header.cupsBytesPerLine);

    ...
  }
}
```

See Also

cupsRasterClose(), cupsRasterOpen(), cupsRasterReadHeader(), cupsRasterWriteHeader(), cupsRasterWritePixels()

cupsRasterWriteHeader()

Usage

```
unsigned
cupsRasterWriteHeader(cups_raster_t *ras,
                      cups_page_header_t *header);
```

Arguments

Argument	Description
ras	The raster stream to write to.
header	A pointer to the page header to write.

Returns

1 on success, 0 on error.

Description

`cupsRasterWriteHeader()` writes the specified page header to a raster stream.

Example

```
#include <cups/raster.h>

int                   line;
cups_raster_t         *ras;
cups_raster_header_t  header;
unsigned char         pixels[8192];
...

cupsRasterWriteHeader(ras, &header);

for (line = 0; line < header.cupsHeight; line ++)
{
  ...

  cupsRasterWritePixels(ras, pixels, header.cupsBytesPerLine);
}
```

See Also

cupsRasterClose(), cupsRasterOpen(), cupsRasterReadHeader(), cupsRasterReadPixels(), cupsRasterWritePixels()

cupsRasterWritePixels()

Usage

```
unsigned
cupsRasterWritePixels(cups_raster_t *ras,
                      unsigned char *pixels,
                      unsigned length);
```

Arguments

Argument	Description
ras	The raster stream to write to.
pixels	The pixel data to write.
length	The number of bytes to write.

Returns

The number of bytes written.

Description

`cupsRasterWritePixels()` writes the specified pixel data to a raster stream.

Example

```
#include <cups/raster.h>

int                  line;
cups_raster_t        *ras;
cups_raster_header_t header;
unsigned char        pixels[8192];
...

cupsRasterWriteHeader(ras, &header);

for (line = 0; line < header.cupsHeight; line ++)
{
  ...

  cupsRasterWritePixels(ras, pixels, header.cupsBytesPerLine);
}
```

See Also

cupsRasterClose(), cupsRasterOpen(), cupsRasterReadHeader(), cupsRasterReadPixels(), cupsRasterWriteHeader()

cupsServer()

Usage

```
const char *
cupsServer(void);
```

Returns

A pointer to the default server name.

Description

`cupsServer()` returns a pointer to the default server name. The server name is stored in a static location and will be overwritten with every call to `cupsServer()`

The default server is determined from the following locations:

1. The `CUPS_SERVER` environment variable,
2. The `ServerName` directive in the *cupsd.conf* file,
3. The default host, "localhost".

Example

```
#include <cups/cups.h>

const char *server;

server = cupsServer();
```

See Also

cupsGetPassword(), cupsUser()

cupsTempFile()

Usage

```
char *
cupsTempFile(char *filename,
             int length);
```

Arguments

Argument	Description
filename	The character string to hold the temporary filename.
length	The size of the filename string in bytes.

Returns

A pointer to `filename`.

Description

`cupsTempFile()` generates a temporary filename for the */var/tmp* directory or the directory specified by the `TMPDIR` environment variable.

Example

```
#include <cups/cups.h>

char filename[256];

cupsTempFile(filename, sizeof(filename));
```

cupsUser()

Usage

```
const char *
cupsUser(void);
```

Returns

A pointer to the current username or `NULL` if the user ID is undefined.

Description

`cupsUser()` returns the name associated with the current user ID as reported by the `getuid()` system call.

Example

```
#include <cups/cups.h>

const char *user;

user = cupsUser();
```

See Also

cupsGetPassword(), cupsServer()

httpBlocking()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

httpCheck()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

httpClearFields()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

httpClose()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

httpConnect()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

httpDecode64()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

httpDelete()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

httpEncode64()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

httpError()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

httpFlush()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

httpGet()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

httpGets()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

httpGetDateString()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

httpGetDateTime()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

httpGetField()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

httpGetLength()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

httpHead()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

httpInitialize()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

httpOptions()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

httpPost()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

httpPrintf()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

httpPut()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

httpRead()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

httpReconnect()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

httpSeparate()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

httpSetField()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

httpTrace()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

httpUpdate()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

httpWrite()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

ippAddBoolean()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

ippAddBooleans()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

ippAddDate()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

ippAddInteger()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

ippAddIntegers()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

ippAddRange()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

ippAddRanges()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

ippAddResolution()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

ippAddResolutions()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

ippAddSeparator()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

ippAddString()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

ippAddStrings()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

ippDateToTime()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

ippDelete()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

ippFindAttribute()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

ippLength()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

ippNew()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

ippPort()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

ippRead()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

ippTimeToDate()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

ippWrite()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

ppdClose()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

ppdConflicts()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

pddEmitFd()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

ppdEmit()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

ppdFindChoice()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

ppdFindMarkedChoice()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

ppdFindOption()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

ppdIsMarked()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

ppdMarkDefaults()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

ppdMarkOption()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

ppdOpenFd()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

ppdOpenFile()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

ppdOpen()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

ppdPageLength()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

ppdPageSize()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

ppdPageWidth()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also